Travaux Dirigés sur le langage C n°.5

Cours d'Informatique de Deuxième Année

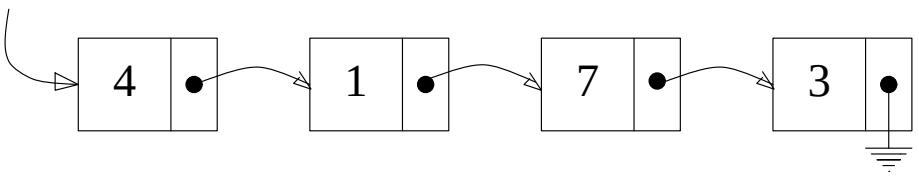
—Deug SSM M3—

Les objectifs de ce *TD* sont les suivants :

- Utilisation de structures et définition de nouveaux types en langage C,
- Étude de la notion de liste chaînée par pointeurs.

Listes chaînées par pointeurs...

Une *liste* est un ensemble de *cellules* "simplement chaînées" : chaque cellule contient principalement les informations que l'on souhaite manipuler mais il faut en plus *lier* toutes ces cellules pour en faire une liste chaînée.



Exemple d'une liste chaînée par pointeurs d'entiers non triés.

Quand nous utilisons des tableaux, nous devons faire de la place ou combler du vide pour respectivement insérer ou supprimer des éléments (voir le cas de la gestion d'un ensemble par tableau trié).

Dans le cas d'un stockage par liste chaînée, ce problème ne se pose plus mais le prix à payer c'est qu'il faut stocker en plus des informations manipulées, "l'adresse" de la cellule suivante dans l'ordre du chaînage — celle-çi était, dans le cas des tableaux, intrinsèquement définie comme la "case suivante" du tableau.

Les définitions de types suivantes mettent en oeuvre les listes chaînées par pointeurs :

```
typedef struct Cellule
{ int Val;                              # donnee stockee :  un entier.
  struct Cellule  * PtrSuiv;            # pointeur sur la cellule suivante.
} StrCellule;                           # definition du nouveau type :  StrCellule.

typedef StrCellule * PtrCellule;        # definition du nouveau type :  PtrCellule.
```

On considère donc des listes chaînées d'entiers : se donner une telle liste, c'est se donner un pointeur sur le premier élément de la liste; ceci motive donc la définition du nouveau type `PtrCellule` (pointeur sur cellule). Afin de simplifier la programmation, on se propose d'implémenter quelques fonctions basiques de manipulation des listes chaînées.

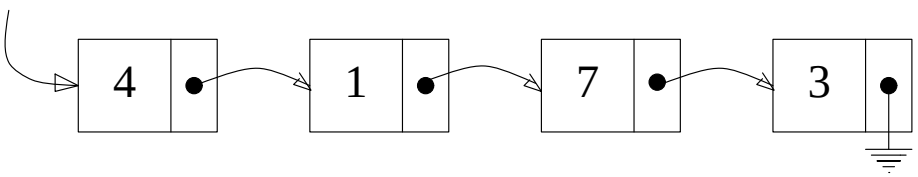
Dans la suite, on parlera de "liste chaînée" pour dénoter un pointeur de type `PtrCellule` sur la première cellule (de type `StrCellule`) d'une liste chaînée par pointeurs.

▸ **Exercice 1**. Écrire une fonction de nom `AlloueCellule` créant une nouvelle cellule `nc` et retournant un pointeur sur `nc`. En déduire la fonction de nom `CreerCellule` prenant en argument un entier `val` et retournant un pointeur sur une nouvelle cellule contenant la valeur `val`. On veillera à ce que le champ `PtrSuiv` de cette structure soit initialisé à la valeur `NULL`.

▸ **Exercice 2**. Écrire une fonction de nom `LitListe` de création d'une liste chaînée d'entiers saisis au clavier. On arrêtera la saisie lorsque l'utilisateur entrera l'entier 0 et on retournera alors un pointeur sur le premier élément de la liste ainsi créée.

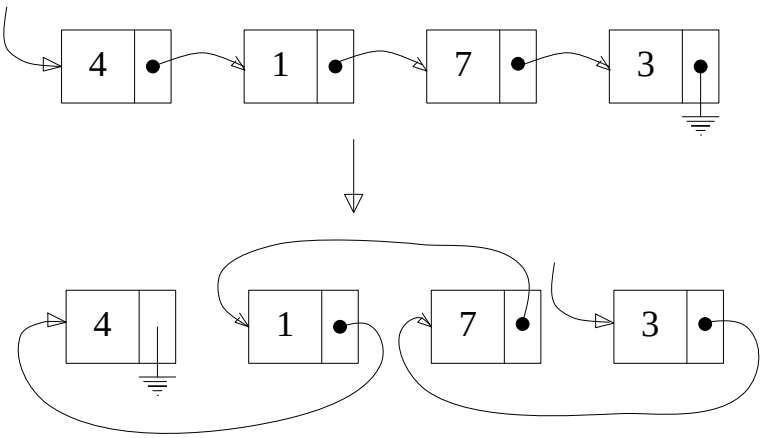
▸ **Exercice 3**. Écrire une fonction de nom `AfficheListe` prenant en argument une liste chaînée référencée par le pointeur `PtrDebut`. La fonction affichera les éléments de la liste dans l'ordre du chaînage.

▸ **Exercice 4**. Écrire une fonction de nom `AfficheListeInv` prenant en argument une liste chaînée référencée par le pointeur `PtrDebut`. La fonction affichera les éléments de la liste dans l'ordre inverse du chaînage. L'exécution sur la liste



devra produire la sortie : 3 7 1 4.

▸ **Exercice 5**. Écrire une fonction de nom `DuplicListe` prenant en argument une liste chaînée référencée par le pointeur `PtrDebut`. La fonction retournera une liste chaînée (totalement indépendante de la liste originale) qui sera une réplique exacte de la liste chaînée de départ.

▸ **Exercice 6**. Écrire une fonction de nom `Inverse` prenant en argument une liste chaînée référencée par le pointeur `PtrDebut`. La fonction inversera l'ordre des éléments de la liste chaînée en créant une nouvelle liste chaînée totalement indépendante de la liste d'origine. Elle ne devra surtout pas détruire le chaînage de la liste originale.

▸ **Exercice 7**. Écrire une fonction de nom `InverseSurPlace` prenant en argument une liste chaînée référencée par le pointeur `PtrDebut`. La fonction inversera l'ordre des éléments de la liste chaînée "sur place". Le chaînage initial sera donc détruit.